Parallel Implicit Solution of Diffusion-limited Radiation Transport

W. D. Gropp[1], D. K. Kaushik[1], D. E. Keyes[2], and B. F. Smith[1]

[1] MCS Div., Argonne National Laboratory
{gropp,kaushik,bsmith}@mcs.anl.gov
[2] APAM Dept., Columbia University david.keyes@columbia.edu

Abstract

We present simulations of diffusion-limited transport in an initially cold medium of two different materials subjected to an impulsive radiative load, using a Newton-Krylov-Schwarz solver. The spatial discretization employs Galerkin finite elements with linear piecewise continuous basis functions over simplices in 2D and 3D. Temporal integration is via a solution-adaptive implicit Euler method. The code shows excellent domain-decomposed scalability on the Teragrid, BlueGene, and System X platforms. Comparing implementations for this application with flop-intensive residual evaluation, we observe that an analytical Jacobian gives better performance (in terms of the overall execution time to solution) than a Jacobian-free approach.

1 Diffusion-limited radiation transport

Under the assumptions of isotropic radiation with no frequency dependence, transport through a material characterized by spatially varying atomic number Z and thermal conductivity of κ can be modeled by the following coupled nonlinear equations, known as flux-limited radiation diffusion [9]:

$$\frac{\partial E}{\partial t} - \nabla \cdot (D_E \nabla E) = \sigma_a(T^4 - E), \quad \frac{\partial T}{\partial t} - \nabla \cdot (D_T \nabla T) = -\sigma_a(T^4 - E) \quad (1)$$

with

$$\sigma_a = \frac{Z^3}{T^3}, \quad D_E(E,T) = \frac{1}{3\sigma_a + \frac{|\nabla E|}{|E|}}, \quad \text{and } D_T(T) = \kappa T^{\frac{5}{2}}. \quad (2)$$

Here, E represents the photon energy density and T is the material temperature. Since the diffusion approximation can allow speeds of propagation faster than speed of light, the above formula for diffusivity D_E includes Wilson's flux limiter $|\nabla E|/|E|$ [9], a strongly nonlinear effect. Though simple in appearance, this is a challenging problem when atomic number varies sharply, due to the cubic dependence of the source term coefficient on Z.

2 Discretization and algorithmic setting

We employ the Galerkin finite element method using conforming linear P1 elements, triangular for 2D and tetrahedral for 3D [5]. The diffusion coefficients, D_E and D_T, are also expanded in the element basis functions.

In this paper, due to space constraints, we present results from backward Euler time integration only. Comparisons with various higher order time integration methods, including BDF (discussed for radiation diffusion problems in [3]) and implicit Runge-Kutta schemes will be published elsewhere. We evolve the timestep size by limiting the changes in the solution (point by point) according to [10]:

$$\max\left(\frac{|U^{n+1} - U^n|}{|U^{n+1}|}\right) \leq \epsilon_t. \tag{3}$$

We have used $\epsilon_t = 0.75$ in all of the computational results in this paper. We start with a small value of δt $(= 10^{-5})$ and allow it to evolve with Eqn. (3), except that the timestep grows by no more than 20% per timestep, and an occasional "short" step is imposed to archive the solution for visualization at regularly spaced intervals. The timestep size evolves to about four orders of magnitude larger than the initial timestep size towards the end of the computation, after the radiation pulse has passed beyond the high atomic number zone.

We use the Newton-Krylov-Schwarz (NKS) algorithm [4, 7] to solve the nonlinear problem arising on every timestep of the discretized form of Eqn. (1). Several parameters of NKS must be tuned for optimal performance [4]. Our code is built on PETSc [1]. We use a left-preconditioned inexact Newton method to solve the nonlinear problem on each timestep. The relative tolerance for the nonlinear residual norm reduction in each timestep is 10^{-8}, which is far below discretization error but within easy reach of Newton's method. The linear problems within each Newton step are solved using GMRES with a maximum of 80 iterations and a maximum subspace size of 30 between restarts, or a relative reduction of the left-preconditioned residual by three orders of magnitude. We use a block Jacobi (zero overlap) preconditioner and map each subdomain to a single processor. Though not algorithmically scalable for general elliptic problems, this inexpensive limit of Schwarz preconditioning is adequate for transient problems. We use incomplete factorization (ILU) within each subdomain and allow a single level of fill. This tuning of the NKS method follows [4], where it was effective in overall runtime for a CFD code on a variety of message passing architectures.

3 Results and Discussion

We present 2D and 3D test cases. The computational domain in 2D is the unit square, with a radiation flux incident on the left boundary. The atomic number is location dependent:

$$Z(x,y) = \begin{cases} 10 \text{ for } \frac{1}{3} \le x \le \frac{2}{3} \text{ and } \frac{1}{3} \le y \le \frac{2}{3}, \\ 1 \quad \text{elsewhere.} \end{cases} \tag{4}$$

The boundary conditions for the Eqns. (1) are set by imposing a constant radiation field at $x = 0$:

$$\begin{aligned} \mathbf{n} \cdot D_E \nabla E + \frac{E}{2} = 2 \text{ at } x = 0 \text{ and } \mathbf{n} \cdot D_E \nabla E + \frac{E}{2} = 0 \text{ at } x = 1, \\ \text{and } \mathbf{n} \cdot \nabla E = 0 \text{ at } y = 0 \text{ and } y = 1, \end{aligned}$$

where $\mathbf{n}$ is the outward unit normal to the boundary, as in [8].

Figure 1 plots the material temperature along $y = 0.5$ and $x = 0.5$ cuts for a large range of uniform mesh resolutions, showing asymptotic grid independence. A sufficiently fine mesh is needed to resolve the sharp features inside and around the interior domain of high atomic number. Time evolution of material temperature along the same cuts is presented in Figure 2. The high-Z region at the center of the interior domain takes longest to heat up.

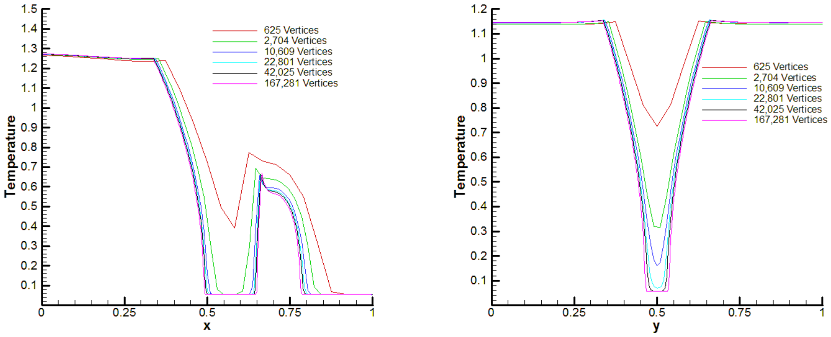

Fig. 1. Material temperature at $y = 0.5$ (left) and at $x = 0.5$ (right) showing mesh independence for the 2D test example at $t = 3$.

The temperature contours showing the propagation of the thermal front from $t = 1$ to $t = 4$ are shown in Figure 3 for the 3D case with a tetrahedral mesh of 237,160 vertices and 1,264,086 elements. This reduces to the 2D case on z = constant planes, as atomic number depends only on the x and y coordinates. This permits comparison to the 2D solution, while providing a fully 3D configuration for demonstrating scaling.

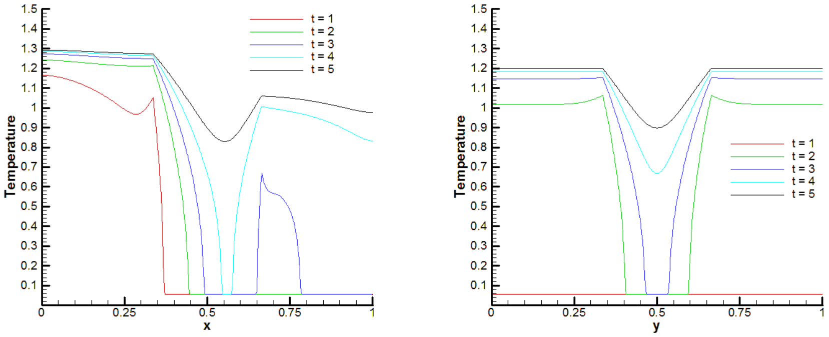

Fig. 2. Material temperature evolution at $y = 0.5$ (left) and at $x = 0.5$ (right) for the 2D test case with 167,281 vertices and 332,928 elements.

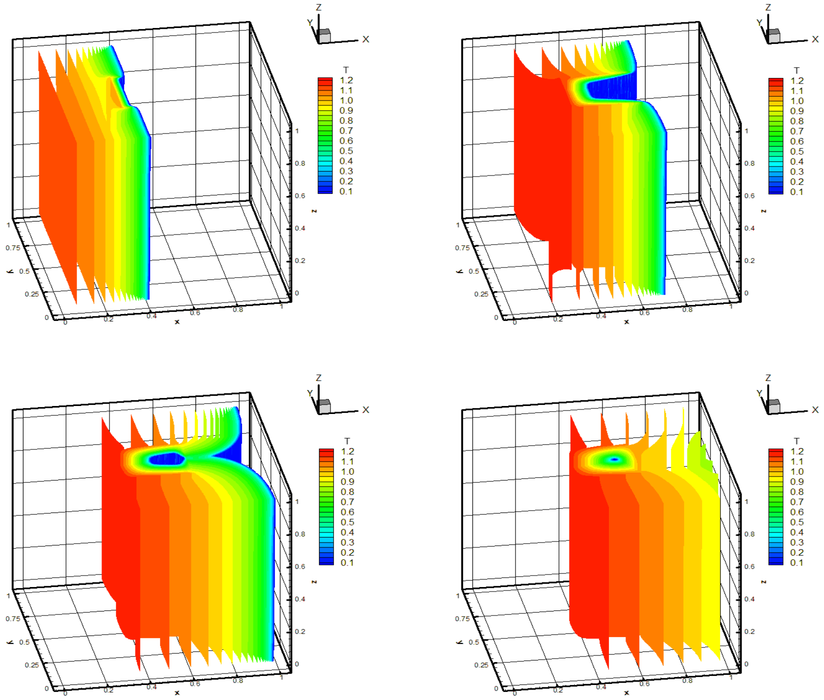

Fig. 3. Evolution of material temperature in time for the 3D test example with a tetrahedral mesh of 237,160 vertices and 1,264,086 elements. The top left figure shows the temperature contours at $t = 1$, the top right at $t = 2$, the bottom left at $t = 3$, and the bottom right at $t = 4$.

We discuss performance issues of the time-accurate NKS algorithm on the Teragrid cluster (SDSC), BlueGene (Argonne), and System X (Virginia Tech). The Teragrid cluster is made of 1.5 GHz dual Intel Madison processors, each with 4 MB of L2 cache, and 4 GB of memory per node. The IBM BlueGene node contains dual 700 MHz processors with 4 MB shared L3 cache and 512 MB of main memory. We do all the computations on BlueGene in the "co-processor" mode, in which one processor of a node does communication only, since memory and memory bandwidth are too limited to make effective use of both processors as floating point engines. System X is a cluster of dual 2.3 GHz PowerPC 970FX (Apple Xserve G5) processors with 0.5 MB L2 cache. We do not discuss here the per-processor floating point performance, though it is an important part of performance overall [4]. Though our code scales well, it will require, as in [4], attention to ordering and blocking issues to achieve a high percentage of machine peak on cache-based microprocessors.

3.1 Algorithmic Performance

Figure 4 shows the history of timestep size evolution according to Eqn. 3 and the average number of linear iterations per timestep. Overall execution time is highly sensitive to timestepping strategy for this problem. We are evaluating higher order (BDF [2] and Implicit Runge Kutta [6]) methods from standpoints of computational efficiency and robustness. However, scalability is not highly sensitive to choice of timestepping scheme, so the results of this study on domain-decomposed preconditioning, which cover a wide range of degree of diagonal dominance in the implicit operator, are representative. In the right side plot of Figure 4, the number of linear iteration count rises as the number of subdomains increases since the block Jacobi preconditioner gets weaker as diagonal dominance diminishes for larger timesteps.

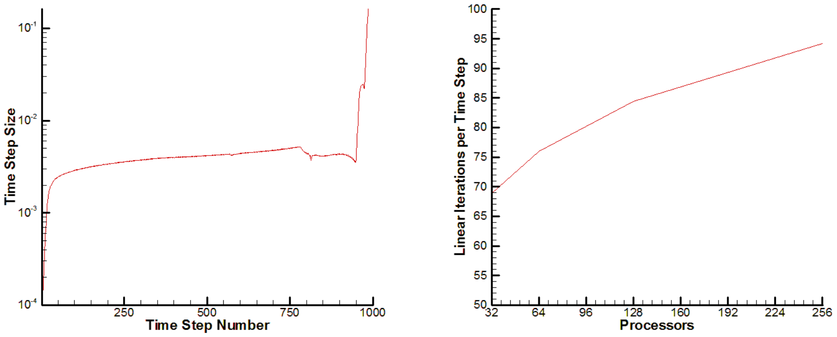

Fig. 4. Time step size history (left) and the average number of linear iterations per timestep (right) for the 3D test case of Figure 3.

3.2 Parallel Scalability

We present the execution time and parallel efficiency on up to 256 processors of Teragrid, BlueGene, and System X in Figure 5. The base case for computing the parallel efficiency is chosen such that the problem fits *comfortably* in the available distributed memory. The code shows excellent scaling on the three machines. The superlinear speedup on the Teragrid cluster is primarily due to the superior cache performance offsetting the increased communication time as the problem size per-processor gets smaller. However, this also shows that we have room for doing more per-processor (memory hierarchy) performance optimizations that will benefit the base case the most. We are currently investigating these performance issues.

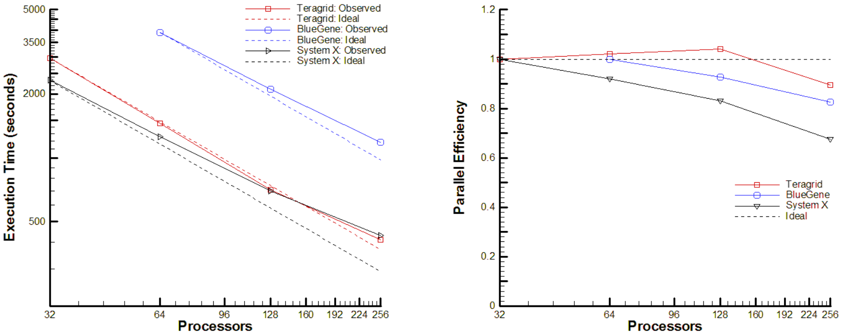

Fig. 5. Execution time (left) and parallel efficiency (right) on 256 processors of Teragrid (1.5 GHz Intel Madison dual processors nodes), BlueGene (700 MHz dual processor nodes), and System X (dual 2.3 GHz PowerPC 970FX processor nodes).

3.3 Analytical vs. Jacobian-free NKS

The diffusion-limited radiation transport equation presents challenging nonlinear behavior. At the same time, it is easy to code the analytical Jacobian (which is used for the preconditioning matrix, as well). The analytical Jacobian has the advantage of possibly better performance (see Table 1) but requires memory to store the matrix explicitly. Another convenient approach is to perform the matrix-vector products (as needed for the Krylov solver) without explicitly forming the Jacobian matrix [7]. This has the obvious advantage of savings in the memory requirements (though the preconditioner matrix may still need to be stored) but requires extra function evaluations. This will compete well with the analytical Jacobian case only when the execution time for function evaluation is significantly (perhaps an order of magnitude) less than the time to compute the analytical Jacobian

In Table 1, we compare the performance of three choices for the time-accurate NKS algorithm:

- **analytical Jacobian** computed to the same order of accuracy as the function in every nonlinear iteration. The preconditioner matrix is chosen to be the same as the Jacobian matrix. The computational cost of this part dominates the execution time: 62% on 256 processors of Teragrid.
- **Jacobian-free** matrix-vector products performed without explicitly forming the Jacobian matrix. However, this matrix (or a cheap approximation to it) is often needed for preconditioning purpose in every nonlinear iteration. The dominating cost here is the function evaluation (about 48% in Table 1)
- **lagged Jacobian-free** matrix-vector products performed the same way as in the previous item but preconditioner matrix is built only once per timestep and reused for all the linear solves with in a timestep. This saves time spent on the preconditioner evaluation but often requires more linear and nonlinear iterations (and thus function evaluations) than the previous choice. One can even freeze the preconditioner evaluation for many timesteps but it should be done only when the step size is small or when there is little change from one step to the next.

We observe that the analytical Jacobian does the best among the three choices in terms of the total wallclock time. The ratio of the cost of one Jacobian evaluation to that of one function evaluation on 256 processors of Teragrid is about thirteen while there is sixteenfold increase in the number of function evaluations for Jacobian-free case (as compared to the analytical Jacobian case). Therefore, the Jacobian-free approach is not competitive in the present scenario even if we assume that the time spent on the preconditioner evaluation is negligible (which may not be the case). However, a short code development cycle and savings in memory must often be considered while choosing Jacobian-free versus analytical approaches.

4 Conclusions and future work

The time-accurate NKS algorithm scales well on Teragrid, BlueGene, and System X platforms. However, the per-processor performance needs memory hierarchy optimizations. This might make the function evaluation phase relatively cheaper, which in turn, can make the Jacobian-free approach competitive. Higher order time integration poses more difficult nonlinear systems by allowing larger timesteps, but they can be more computationally efficient overall by completing the time marching in fewer steps. Future work will also address more complex computational geometries with irregularly shaped zones of different atomic number, as often encountered in practice.

Table 1. Performance comparison of analytical Jacobian, Jacobian-free, and lagged Jacobian-free NKS methods. In the lagged case, Jacobian is evaluated only once per timestep. For the 3D test of Figure 3 on 256 processors of Teragrid.

Number of	Analytical	Jacobian Free	Lagged Jacobian-free
Time Steps	986	986	986
Nonlinear Iter	4,812	4,812	5,778
Linear Iter	92,843	92,842	92,016
Function Eval	6,140	98,982	99,146

Execution Time of	Analytical	Jacobian Free	Lagged Jacobian-free
Function Eval	25	395	396
Jacobian Eval	254	0	0
PC Eval	0	254	52
Total	412	823	601

5 Acknowledgments

We thank Dimitri Mavriplis of the Univ. of Wyoming and Paul Fischer of Argonne Nat. Lab. for many helpful discussions. Computer time was supplied by Argonne Nat. Lab., by NSF's Teragrid facility, and by Virginia Tech.

References

1. S. Balay, K. R. Buschelman, W. D. Gropp, D. K. Kaushik, M. G. Knepley, L. C. McInnes, and B. F. Smith. PETSc home page. `http://www.mcs.anl.gov/petsc`, 2002.
2. P. N. Brown, G. D. Byrne, and A. C. Hindmarsh. A variable-coefficient ODE solver. *SIAM Journal of Sci. Statist. Comput.*, 10:1038–1051, 1989.
3. P. N. Brown, D. E. Shumaker, and C. Woodward. Fully implicit solution of large-scale non-equilibrium radiation diffusion with high order time integration. *Journal of Comp. Phys.*, 204:760–783, 2005.
4. W. D. Gropp, D. K. Kaushik, D. E. Keyes, and B. F. Smith. High performance parallel implicit CFD. *Journal of Parallel Computing*, 27:337–362, 2001.
5. Thomas J. R. Hughes. *The Finite Element Method.* Dover Publications, Inc., Mineola, New York, 2000.
6. G. Jothiprasad, D. J. Mavriplis, and D. A. Caughey. Higher order time integration schemes for the unsteady Navier-Stokes equations on unstructured meshes. In *Proc. of 32nd AIAA Fluid Dynamics Conference*, 2002.
7. D. A. Knoll and D. E. Keyes. Jacobian-free Newton-Krylov Methods: A survey of approaches and applications. *Journal of Comp. Phys.*, 193:357–397, 2004.
8. D. J. Mavriplis. Multigrid approaches to non-linear diffusion problems on unstructured meshes. *Numerical Linear Algebra with Applications*, 8:499–512, 2001.
9. D. Mihalas and B. Weibel-Mihalas. *Foundations of Radiation Hydrodynamics.* Dover Publications, Inc., Mineola, New York, 1999.

10. M. Pernice and B. Philip. Solution of equilibrium radiation diffusion problems using implicit adaptive mesh refinement. *SIAM Journal of Sci. Comput.*, 27(5):1709–1726, 2006.